WHAT IS CLAIMED IS:

1. A method for testing operation of a system against a natural language design specification defining a closed set of behaviors thereof, the system being operationally specific to at least one problem domain, the method comprising the steps of:

translating the natural language specification to a specification in at least one description language specific to a corresponding one of the at least one problem domain, each of said at least one domain specific description language including a set of axioms, each of said set of axioms corresponding to a respective syntactical relationship between morphemes of said corresponding domain specific description language;

generating a model of the system from said domain specific description language specification, said model behaving in accordance with a first set of behaviors and a second set of behaviors, each of said first set of behaviors corresponding to a respective one of the closed set of behaviors and each of said second set of behaviors corresponding to a respective one of said set of axioms;

generating from said model a test case for each behavior of a set union of said first set of behaviors and said second set of behaviors; and

providing to the system every test case generated from said model as the set of test cases for testing the system operation.

2. The method for testing operation of a system by providing a set of test cases thereto as recited in Claim 1, whereby said model includes an extended finite state machine.

3. The method for testing operation of a system by providing a set of test cases thereto as recited in Claim 2, whereby at least one state transition corresponding each of said axioms is embedded into said extended finite state machine.

4. The method for testing operation of a system by providing a set of test cases thereto as recited in Claim 1, whereby each of said at least one domain specific description language is a respective domain specific computer programming language.

5. The method for testing operation of a system by providing a set of test cases thereto as recited in Claim 4, whereby each of said domain specific programming language is a respective functional language.

6. The method for testing operation of a system by providing a set of test cases thereto as recited in Claim 5, whereby said model generating step includes the steps of:

reordering functions in said domain specific programming language specification;

separating said functions into a first set of functions corresponding to functions native to said domain specific programming language and a second set of functions corresponding to functions foreign to said domain programming specific language;

generating an extended finite state machine from said second set of functions; and

embedding into said extended finite state machine at least one state transition corresponding to each of said first set of functions.

7. The method for testing operation of a system by providing a set of test cases thereto as recited in Claim 6, whereby said functions are reordered in a temporal order of execution.

8. The method for testing operation of a system by providing a set of test cases thereto as recited in Claim 6, whereby said first set of functions includes recursion.

9. The method for testing operation of a system by providing a set of test cases thereto as recited in Claim 8, whereby a state transition is embedded into said extended finite state machine for each of a base case, a terminating case and one recursive step of said recursion function.

10. A method for testing a computing application specific to a computing domain comprising the steps of:

providing to a testing authority a natural language specification of the computing application defining a closed set of behaviors thereof;

translating said natural language specification into a specification in at least one computing language specific to the computing domain;

generating a model of the application from said at least one domain specific language specification, said model behaving in accordance a set union of behaviors corresponding to at least one set of axioms derived from said at least one domain specific language and said closed set of behaviors;

generating a set of test cases executable by the computing application from said model, each of said set of test cases corresponding to one of said set union of behaviors;

executing each of said set of test cases by means of the computing application; and

communicating to said testing authority each of said set of test cases which results in a behavior of the application contrary to said corresponding one of said closed set of behaviors specified in said natural language specification.

11. The method for testing a computing application as recited in Claim 10, whereby each of said at least one domain specific language is a functional language.

12. The method for testing a computing application as recited in Claim 10, whereby all of said at least one domain specific language are derived from a common functional language.

13. The method for testing a computing application as recited in Claim 12, whereby said common functional language is Haskell.

14. The method for testing a computing application as recited in Claim 13, whereby one of said at least one domain specific language is HaskellDB.

15. The method for testing a computing application as recited in Claim 14, whereby one of said at least one domain specific language is TclHaskell.

16. The method for testing a computing application as recited in Claim 13, whereby one of said at least one domain specific language is TclHaskell.

17. The method for testing a computing application as recited in Claim 10, whereby said behavioral model includes an extended finite state machine defined by said closed set of behaviors.

18. The method for testing a computing application as recited in Claim 17, whereby said extended finite state machine is embedded with state transitions corresponding to each of said at least one set of axioms.

19. The method for testing a computing application as recited in Claim 11, whereby said model generating step further includes the steps of:

reordering functions in said domain specific language specification;

separating said functions into a first set of functions corresponding to functions native to said domain specific language and a second set of functions corresponding to functions foreign to said domain specific language;

generating an extended finite state machine from said second set of functions; and

embedding into said extended finite state machine at least one state transition corresponding to each of said first set of functions.

20. The method for testing a computing application as recited in Claim 19, whereby said functions are reordered in a temporal order of execution.

21. The method for testing a computing application as recited in Claim 19, whereby said first set of functions includes recursion.

22. The method for testing a computing application as recited in Claim 21, whereby a state transition is embedded into said extended finite state machine for each of a base case, a terminating case and one recursive step of said recursion function.